Getting started with Node.js

Full-stack application:

A full-stack application includes the following components:

The Client Side:

The website and the mobile application that are user-facing.

The Server Side:

Processes any requests from the client side and sends responses back to the client. In today's world, the cloud hosts the web server, application server, and database.

Open Source and Cross-Platform:

JavaScript is an ideal choice for client-side validation of HTML pages due to its versatility and cross-platform compatibility. Recognizing its ease of use, JavaScript was further adapted for server-side coding, resulting in Node.js. Node.js functions as a runtime environment, and what's more, it requires no special licenses for use. Additionally, a plethora of packages and libraries have been developed for use with Node.js, thanks to its open-source nature. Furthermore, Node.js code can seamlessly run across various operating systems such as Linux, Windows, and Mac OSX.

V8 Engine:

Every piece of code you write needs to undergo processing and conversion into a machine-readable form. In the realm of Node.js, JavaScript code is executed using the Google V8 engine. Renowned for its high performance, V8 is an open-source engine developed by Google, and it is integrated into all Google Chrome browsers. Modern browsers like Microsoft Edge use the V8 engine for JavaScript, while Node.js uses V8 on the server side.

Event-Driven, Asynchronous, Non-Blocking, Single-Threaded:

Server processes can be categorized as either "single-threaded" or "multi-threaded". In a single-threaded environment, only one command is processed at any given time, while in a multi-threaded environment, multiple commands can be processed simultaneously. Despite being single-threaded, Node.js excels in performance due to its asynchronous and non-blocking nature. This means that while a process is being executed, the program does not need to wait until it finishes. Node.js is event-driven, meaning that when it performs an input/output (I/O) operation, such as reading from the network or accessing a database or file system, an event is triggered. Instead of blocking the thread and consuming processor time while waiting, Node.js resumes operations when the response is received, or when the corresponding event occurs. This non-blocking behavior enables the server to remain responsive and handle multiple tasks concurrently, akin to a multi-threaded environment.

JSON Payload

JSON stands for JavaScript Object Notation, formatted as "key-value pairs". The payload represents the data transmitted between the client and the server. When the client needs to send data to the server, it does so in the form of a JSON object, as illustrated in the example below:

```
{
    "name": "John",
    "age": "24",
```

```
    "email": "johnparker@gmail.com"
}
```

The above object is a JSON. When this data is sent as part of the request, its values can be extracted simply by using `request.name` and so on.

Similarly, the response is also sent as JSON. An example is provided below:

```
{
    "status": "ok",
    "message": "Successfully added"
}
```

Express Framework

While Node.js provides packages to create a server, the Express framework simplifies the process of creating APIs and endpoints. An API endpoint is a specific point of entry for a request from the client to the server.

Next steps

In this course, you will learn how to perform server-side coding with Node.js using the Express framework.